National Taiwan University
Department of Electrical Engineering
Name: Ryan Hsiang
ID: B11901040

113-1 (Fall 2024)
Reinforcement Learning

Assignment 3 Report

Q1. Try using at least two different algorithms to train your 2048 model. Present your results and discuss possible reasons why some algorithms perform better than others in this task. (12 pts.)

I tried using DQN and PPO for training. For DQN, I also worked with Double-DQN to lower the maximization bias. Here is the training curve:

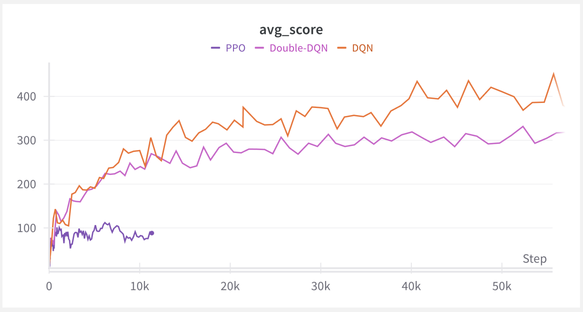

Figure 1: Training curve of average score

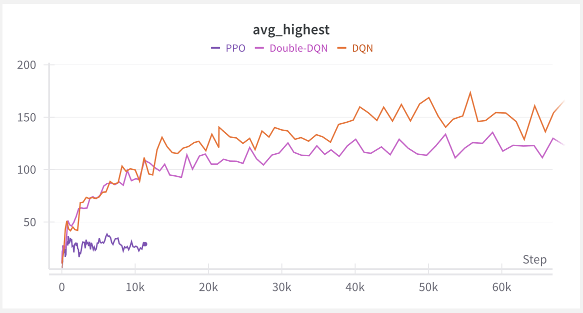

Figure 2: Training curve of average highest

PPO performs significantly worse than both DQN and Double-DQN, while Double-DQN is slightly worse than DQN. PPO is based on policy gradient methods, and is more suitable for continuous action spaces. However, in 2048, the action space is discrete and only of size 4, which is why PPO struggles to learn well. Double-DQN are more stable than DQN, which may lead to a slower convergence process. On the other hand, DQN can learn quickly in simple environments like 2048 due to its overestimation of future rewards.

Q2. Show your best tile distribution in 2048. (4 pts.)

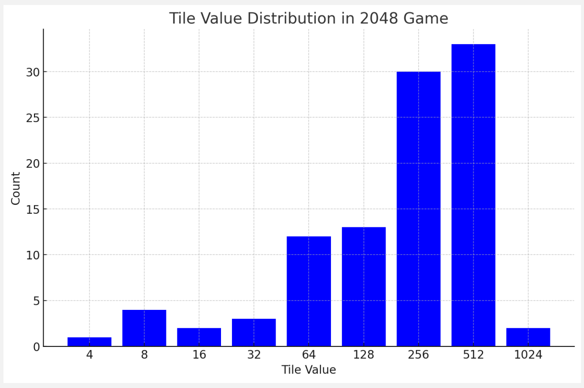

Figure 3: Tile distribution

Q3. Describe what you have done to train your best model. (10 pts.)

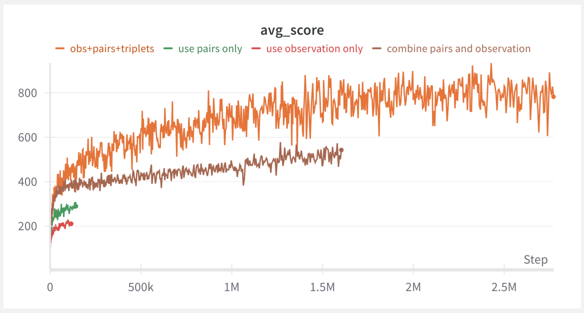

Figure 4: Training curve for different features

I designed a custom feature extractor to compute the one-hot encoding for adjacent tile pairs. There are a total of 24 pairs on a 4×4 board. Each can contain 16×16 possible values, so an additional $24 \times 16 \times 16 = 6144$ values is use as a feature. In addition, I also computed the one-hot encoding for adjacent triplets, for an additional $16 \times 16 \times 16 \times 16 = 65536$ features. The pairs and triplets are concatenated with the original observations to form an input feature of size 71936. These features are then fed into a two layers of MLP, each with 64 neurons. For the reward, I experimented with different reward signals, but found that using the number of empty tiles as reward seemed to be simplest and most effective. This resulted in the agent reaching 1024 by about 300k steps.

Q4. Choose an environment from the Gymnasium library and train an agent. Show your results or share anything you like. (10 pts.)

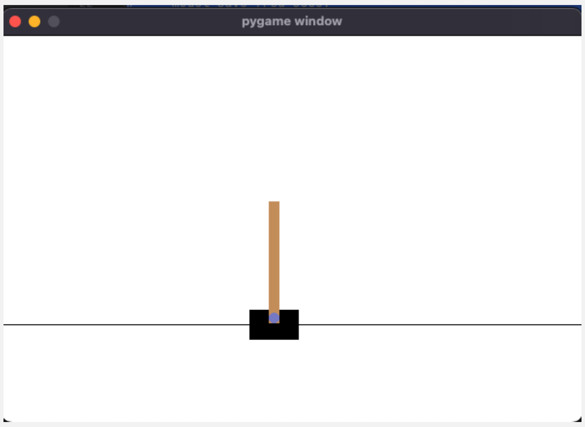

Figure 5: Cartpole environment

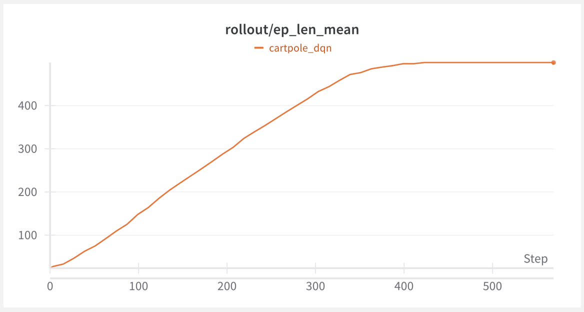

Figure 6: Episode length curve

I attempted to train the agent on the cartpole-v1 environment provided by the Gymnasium library using DQN. I initially trained for only 10000 steps, but the agent quickly tipped over the pole. However, after training for 100000 steps, the agent is able to balance the pole, but found that it moved out of bounds after about 30 seconds. Training for 500000 steps didn't solve this problem, so I decided to use PPO instead for 100000 steps, and found that it performed much better, got much higher reward, and is able to balance the pole while barely moving horizontally. This is probably because PPO is policy-based, and thus are better suited for environments where continuous adjustments are needed, such as balancing a cartpole.